

Delphi Vordiplomsprogramm

Enigma

von Robert Räthel

- Verbesserung -

Fachrichtung : Medieninformatik

Matrikelnummer : 2121

Verwaltungs- / Fachsemester : 4

Email : Robert-Raethel@KreativDuo.net

Inhaltsverzeichnis

I.	Benutzerhandbuch	3.
1.	Ablaufbedingungen	3.
2.	Programminstallation	3.
3.	Programmstart	3.
4.	Bedienungsanleitung	4.
4.1	Hintergrundinformation	4.
4.2	Die Enigma (Übersicht)	5.
4.3	Das Eingabefeld	5.
4.4	Das Ergebnisfeld	5.
4.5	Rotoren zurücksetzen	6.
4.6	Die Einstellungen (Übersicht)	6.
4.6.1	Plugboard ändern	6.
4.6.2	Rotoren ändern	7.
4.6.3	Umkehrwalze ändern	7.
4.6.4	Übernehmen	8.
4.6.5	Abbrechen	8.
4.6.6	Sichern	8.
4.6.7	Laden	8.
4.7	Datei Konvertieren	8.
4.8	Voreinstellungen konfigurieren	9.
5.	Wiederanlaufbedingungen	9.
6.	Fehlermeldungen	10.
II.	Programmierhandbuch	11.
1.	Entwicklungskonfiguration	11.
2.	Problemanalyse und Realisation	11.
2.1	Problemanalyse	11.
2.1.1	Hauptformular	11.
2.1.2	Formular Einstellungen	11.
2.2	Realisierungsanalyse	12.
2.2.1	Datei konvertieren	12.
2.2.2	Plugboard	12.
2.2.3	Steckplätze	13.
2.2.4	Rotoren	13.
2.3	Realisierungsbeschreibung	14.
2.3.1	Grundsätzliches	14.
2.3.2	Programmabläufe	14.
	Programm-Start	
	Programm-Ende	
	Datei konvertieren	
	Formular „Einstellungen“ öffnen	
	Plugboard ändern	
	Einstellungen speichern	
	Ringe initialisieren	
2.3.3	Chiffrierung	16.
2.4	Memory Check (Übersicht)	17.
3.	Datenstrukturen	18.
3.1	Einstellungen	18.
3.2	Rotoren bzw. Ringe	18.
4.	Programmorganisationsplan	19.
5.	Programmtests	20.
6.	Änderungen...	23.
6.1	...am Quellcode	23.
6.2	...an der Dokumentation	23.

I. Benutzerhandbuch

1. Ablaufbedingungen

CPU	: mind. PentiumI mit 200 MHz
Arbeitsspeicher	: mind. 32 MB
Monitor	: 17" oder 19" – Monitor
Auflösung	: 1024 x 768px (optimal)
Betriebssystem	: Windows XP, 2000, 98SE

2. Programminstallation

Die Enigma braucht nicht installiert zu werden. Es ist lediglich notwendig den Ordner „Enigma“ von der beiliegenden CD auf die Festplatte zu kopieren.

Legen Sie dazu die CD in das CD-Rom-Laufwerk Ihres Computers. Öffnen Sie den Arbeitsplatz und wählen Sie Ihr CD-Rom-Laufwerk an. Klicken Sie mit der rechten Maustaste auf den Ordner „Enigma“. Es öffnet sich ein Kontext-Menü. Wählen Sie den Eintrag „Kopieren“. Schließen Sie den Arbeitsplatz wieder. Klicken Sie nun mit der rechten Maustaste auf Ihren Desktop. Wieder öffnet sich ein Kontext-Menü. Wählen Sie hier den Eintrag „Einfügen“. Der Ordner wird auf die Festplatte kopiert. Nach dem Kopiervorgang ist die Installation abgeschlossen.

3. Programmstart

Machen Sie zunächst mit der linken Maustaste einen Doppelklick auf den Ordner „Enigma“. Der Ordner öffnet sich. Doppelklicken Sie nun mit der linken Maustaste auf die Datei „Enigma.exe“. Das Programm öffnet sich.

Hinweis:

Es wird beim ersten Programmstart eine Datei mit dem Dateinamen „Enigma.ini“ in demselben Ordner angelegt. Mithilfe dieser Datei können Sie die Voreinstellungen der Enigma konfigurieren. Lesen Sie dazu aber unbedingt das Kapitel „Voreinstellungen konfigurieren“.

4. Bedienungsanleitung

Bitte lesen Sie diese Bedienungsanleitung sorgfältig, bevor Sie mit dem Verschlüsseln von Texten beginnen.

4.1 Hintergrundinformationen

Die Enigma ist eine Codierungsmaschine aus dem 2. Weltkrieg. Sie wurde verwendet, um Nachrichten zu verschlüsseln und wieder zu entschlüsseln. Bei der Codierung, und gleichermaßen bei der Decodierung, durchläuft ein eingegebener Buchstabe verschiedene Verschlüsselungseinheiten. Die folgende Grafik veranschaulicht Ihnen die Funktionsweise der einzelnen Komponenten. Zur Erklärung gibt es in diesem Beispiel folgende Symbole: ♣, ♠, ♥ und ♦. Es wird ein ♣ verschlüsselt.

Ablauf einer Chiffrierung:

1. **Plugboard:**

Ein Plugboard kann Symbole miteinander vertauschen. In diesem Beispiel gilt: ♣ = ♥ und ♥ = ♣; die beiden übrigen Symbole bleiben unverschlüsselt.

Geben Sie beispielsweise ♣ ein, dann ist es nach dem Plugboard ein ♥.

2. **Rotor:**

Bei einem Rotor gibt es klar definierte Paare.

Hier gilt: ♣ = ♥ und ♠ = ♦ und ♥ = ♠ und ♦ = ♥.

Das ♥ wird zum ♠.

3. **Umkehrwalze:**

Die Umkehrwalze vertauscht alle Symbole miteinander.

Dadurch wird nie ein Symbol auf sich selbst abgebildet.

Es gilt: ♣ = ♠; ♠ = ♣ und ♥ = ♦; ♦ = ♥.

♠ wird demnach zu ♣.

4. **Rotor:** Nach der Umkehrwalze wird wieder durch den Rotor verschlüsselt.

♣ wird zu ♦.

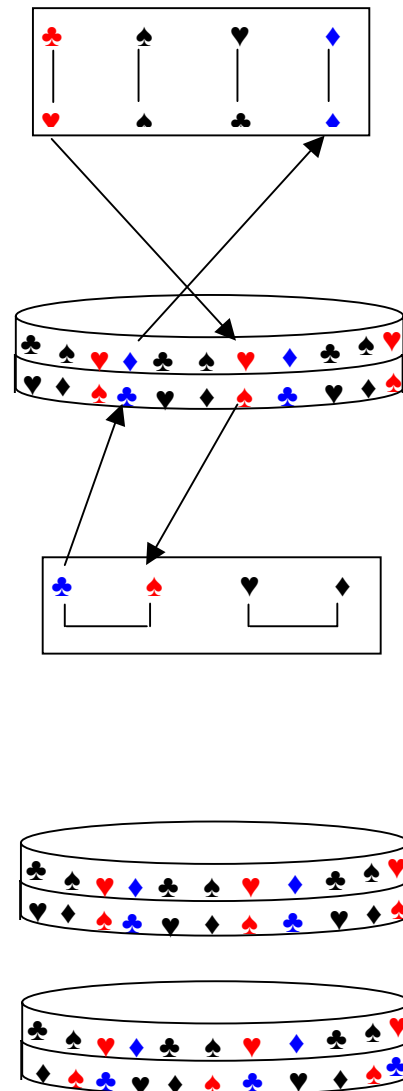
5. **Plugboard:** Zum Schluss wird das Symbol durch das Plugboard chiffriert.

♦ bleibt ♦.

6. Das Symbol ♣ wird also insgesamt zu einem ♦. Als letzten Verschlüsselungs-Trick wird eine Scheibe des Rotors um eine Stelle weitergedreht. Dadurch ergeben sich nach jeder Verschlüsselung neue Paare. Die Codierung eines Symbols ist jetzt von der Anzahl der bereits codierten Symbole abhängig.

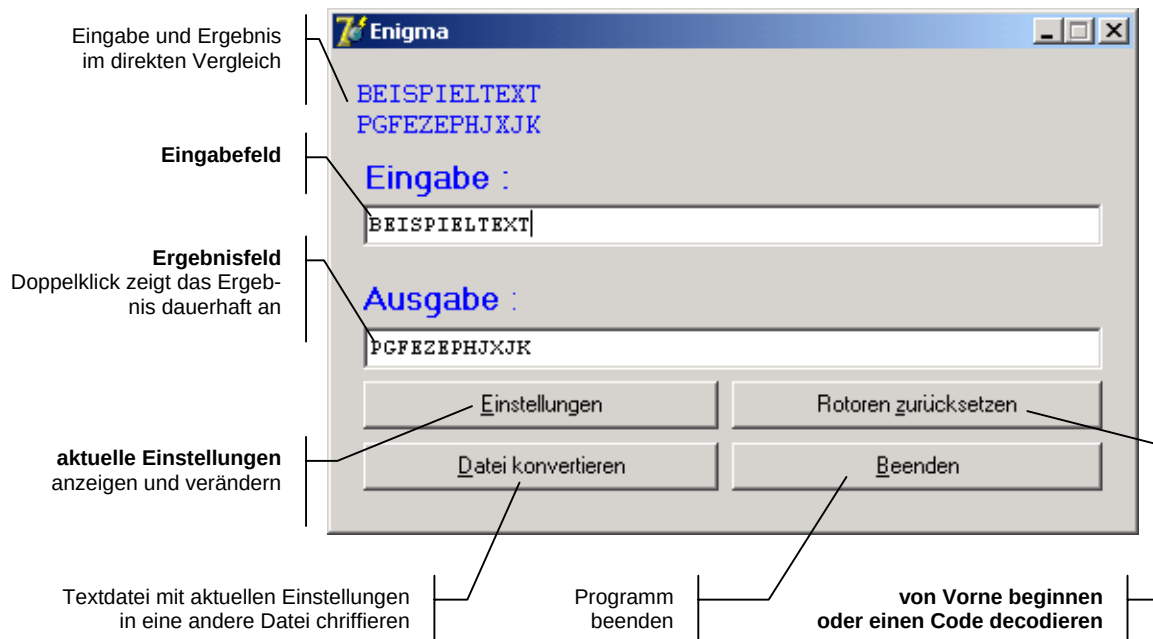
Der Codierungsvorgang ist abgeschlossen.

Bei der Enigma gibt es fünf verschiedene Rotoren, aus denen man drei auswählen kann. Die drei Rotoren werden einfach hintereinander geschaltet. Jeder Rotor besitzt eine beliebig veränderbare Anfangsposition. Das Plugboard ist individuell einstellbar. Ausserdem gibt es zwei Umkehrwalzen, von denen man eine auswählen kann. Alle Verschlüsselungseinheiten der Enigma sind also individuell einstellbar.



4.2 Die Enigma (Übersicht)

Nach dem Programmstart sehen Sie folgendes Fenster:



4.3 Das Eingabefeld

Geben Sie im Eingabefeld einen beliebigen Text ein (z.B. „BEISPIELTEXT“). Sie erhalten sofort die codierte Ausgabe im Ergebnisfeld. Bitte beachten Sie, dass ausschließlich Buchstaben von A bis Z zugelassen sind - alle anderen Tasten sind im Eingabefeld gesperrt. Natürlich können Sie den Cursor bewegen und auch Zeichen löschen, aber berichtigen können Sie auf diese Weise nichts. Der Grund ist Folgender: die Rotoren der Enigma-Maschine drehen sich bei jedem gültigen Buchstaben weiter. Der zuerst eingegebene Buchstabe wird auch als erster Buchstabe codiert. Den zweiten Buchstaben, den Sie eingeben (egal, ob vor oder nach dem Ersten) wird als zweiter Buchstabe codiert. Es wird also genaunommen die Eingabereihenfolge codiert. Aus diesem Grund sehen Sie oberhalb des Eingabefeldes die eingegebenen Buchstaben und den dazugehörigen Code im direkten Vergleich.

Sollten Sie sich vertippt haben, müssen Sie die Eingabe komplett wiederholen. Klicken Sie dazu die Schaltfläche [Rotoren zurücksetzen] an. (Dies gelingt übrigens auch mit Hilfe der Tastatur: Drücken Sie die Tabulatorentaste und dann die Returnntaste.) Die Programmoberfläche und die Rotoren werden zurückgesetzt. Sie können sofort mit der Wiederholung beginnen.

Es werden immer die aktuellen Einstellungen zur Chiffrierung verwendet.

4.4 Das Ergebnisfeld

Im Ergebnisfeld sind alle Tasten gesperrt. Allerdings können Sie mit der Maus die chiffrierten Buchstaben markieren und dann kopieren, um diese zum Beispiel in eine Email einzufügen und dann zu verschicken. Wenn Sie das Ergebnisfeld doppelt mit der linken Maustaste anklicken, werden die Buchstaben über der Ausgabenzeile dauerhaft angezeigt.



4.5 Rotoren zurücksetzen

Um einen Text zu decodieren muss die Enigma-Maschine so eingestellt sein, wie Sie zum Codieren eingestellt war. Anderenfalls erhalten Sie falsche Ergebnisse. Die Einstellungen werden im nächsten Kapitel erläutert. Wenn Sie die Einstellungen nach dem Codieren nicht verändert haben, dann genügt es die Rotoren auf die Anfangstellungen zurückzusetzen. Klicken Sie hierzu auf die Schaltfläche [Rotoren zurücksetzen]. Alle Eingabebefehle sind nun wieder leer. Geben Sie in dem Eingabefeld keinen Text sondern einen Code, wie zum Beispiel: „PGFEZEPHJXJK“, ein. Sie erhalten in dem Ausgabefeld „BEISPIELTEXT“.

4.6 Die Einstellungen (Übersicht)

Ein Klick auf die Schaltfläche [Einstellungen] öffnet folgendes Fenster:

Plugboard
Vertauschte Buchstaben sind farbig hervorgehoben. Graue Buchstaben sind nicht vertauscht. Beispiele:
rot -- A=R; R=A
grau -- F=F
Es können maximal 20 Buchstaben vertauscht sein.

Rotoren
In Steckplatz 1 steckt Rotor I
In Steckplatz 2 steckt Rotor II
In Steckplatz 3 steckt Rotor III

Anfangsstellung
von Rotor I ist A
von Rotor II ist B
von Rotor III ist O

Umkehrwalze
Es wird Umkehrwalze A verwendet. Um die Umkehrwalze zu wechseln.

Reset
Um das Plugboard neu einzustellen

Default
Um die voreingestellten Werte des Plugboards zu laden.

Reset
Um andere Rotoren auszuwählen und eine neue Anfangstellung einzustellen.

Übernehmen
Schließt das Einstellungs-Fenster. Die angezeigten Einstellungen werden geladen.

Abbrechen
Schließt das Einstellungs-Fenster. Die alten Einstellungen werden geladen.

Laden
Einstellungen können aus einer externen Datei eingelesen werden.

Sichern
Die angezeigten Einstellungen können in einer externen Datei abgespeichert werden.

4.6.1 Plugboard ändern

Die Buchstabenfelder des Plugboards sind normalerweise gesperrt. Um die Plugboard-Einstellungen zu ändern, müssen Sie zuerst mit der Maus auf die Schaltfläche [R] wie Reset klicken. Danach sind alle Buchstabenfelder grau hinterlegt und freigegeben. Sie können –müssen aber nicht– bis zu 20 Buchstaben nacheinander anklicken. Jedes so ausgewählte Buchstabenpaar bekommt eine eigene Hintergrundfarbe. Zwei Buchstaben mit gleicher Farbe werden bei der Chiffrierung vertauscht. Grau hinterlegte Buchstaben werden nicht vertauscht.

Sollten Sie sich verlickt haben, müssen Sie auf [R]reset klicken, um von vorne zu beginnen.

Die Schaltfläche [D] steht für Default und lädt bei Bedarf die Voreingestellte-Einstellung aus der ini-Datei.

4.6.2 Rotoren ändern

Auch die Rotoren sind gesperrt, um diese zu verändern müssen Sie die Schaltfläche [RESET] anwählen. Dabei ändert sich die Programmoberfläche. Nebenstehend sehen Sie den Programmausschnitt nachdem auf [RESET] gedrückt wurde. Jetzt müssen Sie nacheinander drei von den fünf Rotoren auswählen. Bitte beachten Sie, dass ein Buchstabe erst durch den Rotor der Stelle1, danach durch den Rotor der Stelle2 und anschließend durch den Rotor der Stelle3 chiffriert wird. Anschließend durchläuft der Buchstabe eine Umkehrwalze und dann die Rotoren der Stelle3, der Stelle2 und anschließend wieder den Rotor der Stelle1. Es macht also einen großen Unterschied, für welchen Rotor Sie sich zuerst entscheiden. Im folgendem Beispiel soll zuerst durch den Rotor II, danach durch Rotor V und anschließend durch Rotor III chiffriert werden.

Um diese Einstellung vorzunehmen müssen Sie die Rotoren in genau dieser Reihenfolge auswählen. Von links nach rechts sehen Sie jeden Zwischenschritt:

Das Programm ist so gestaltet, dass Sie niemals einen Rotor zweimal verwenden können. Sie benutzen Rotor II in der Stelle 1 und damit können Sie den Rotor II nicht mehr in Stelle 2 oder in Stelle 3 benutzen. Deshalb fehlt dort der entsprechende Eintrag.

Die Rotoren müssen noch auf eine Anfangsstellung gesetzt werden. Jetzt, da Sie drei neue Rotoren ausgewählt haben, ist der richtige Zeitpunkt um dies zu tun. Geben Sie einen dreistelligen Buchstabenschlüssel ein (z.B. „OPA“) und bestätigen Sie diesen mit der Return-Taste. Der Rotor in der Stelle1 wird auf „O“ gedreht, der Rotor in der Stelle2 wird auf „P“ gedreht und der dritte Rotor wird folgerichtig auf „A“ gedreht. Erst wenn Sie drei Buchstaben eingegeben haben und diese mit Return bestätigt wurden, sehen Sie nebenstehende Erfolgsmeldung. Und erst dann –und nur dann– hat das Programm Ihre Rotorenwahl akzeptiert und die Anfangstellung gespeichert. Um das Fenster mit der Erfolgsmeldung zu schließen, drücken Sie erneut die Returntaste. Sollten Sie sich vertippt oder verklickt haben, müssen Sie die Schaltfläche [RESET] anklicken und Ihre Rotorenwahl noch einmal von vorne beginnen.

4.6.3 Umkehrwalze ändern

Es ist immer eine der beiden Umkehrwalzen ausgewählt. Möchten Sie die andere Umkehrwalze verwenden, dann klicken Sie diese einfach an.

4.6.4 Übernehmen

Sie haben die Einstellungen individuell verändert und wollen die Einstellungen benutzen. Klicken Sie auf die Schaltfläche [Übernehmen]. Das Programm schließt das Einstellungsfenster und benutzt die neuen Einstellungen. Sie können mit den neuen Einstellungen sofort chiffrieren. Die Schaltfläche [Rotoren zurücksetzen] setzt die Rotoren jetzt auf die neu eingestellten Anfangstellungen zurück.

Sollten die Einstellungen fehlerhaft sein, dann werden Sie mit einer Fehlermeldung darauf hingewiesen. Sie haben keine Möglichkeit die Einstellungen zu berichtigen. Das Fenster mit den Einstellungen schließt sich also grundsätzlich, wenn Sie auf [Übernehmen] klicken.

Es werden die Einstellungen verwendet, die beim Öffnen des Fensters benutzt wurden.

4.6.5 Abbrechen

Sie haben die Einstellungen individuell verändert und interessieren sich nicht mehr dafür – oder sie wollten nur mal nachschauen, welche Einstellungen gerade gelten. Klicken Sie auf die Schaltfläche [Abbrechen]. Das Programm schließt das Einstellungs-Fenster. Die Chiffrierung kann im Hauptprogramm normal fortgesetzt werden.

Es werden die Einstellungen verwendet, die beim Öffnen des Fensters Einstellungen benutzt wurden.

4.6.6 Sichern

Sie haben die Möglichkeit Ihre individuellen Einstellungen in einer Datei zu sichern. Klicken Sie hierzu auf die Schaltfläche [Sichern]. Es öffnet sich ein sogenannter Speichern-Dialog. Sie haben die Möglichkeit in ein beliebiges Verzeichnis Ihres Computers zu wechseln und müssen einen Dateinamen angeben. Die Dateierweiterung von Enigma-Datensätzen ist „.dat“.

Es können verschiedene Fehlermeldungen auftreten. Beachten Sie hierzu bitte das gesonderte Kapitel „Fehlermeldungen“

4.6.7 Laden

Sie haben ebenfalls die Möglichkeit Einstellungen aus einer Datei zu laden. Klicken Sie hierzu auf die Schaltfläche [Laden]. Es öffnet sich ein sogenannter Öffnen-Dialog. Sie haben die Möglichkeit in ein beliebiges Verzeichnis Ihres Computers zu wechseln und müssen eine Datei mit einem gültigen Enigma-Datensatz auswählen. Die Dateierweiterung von Enigma-Datensätzen ist „.dat“. Das bedeutet nicht, dass jede Datei, welche auf .dat endet eine gültige Enigma Datei ist.

Es können verschiedene Fehlermeldungen auftreten. Beachten Sie hierzu bitte das gesonderte Kapitel „Fehlermeldungen“.

4.7. Datei konvertieren

Die Schaltfläche [Datei konvertieren] ermöglicht Ihnen den Inhalt einer Textdatei zu codieren und diesen Code in eine andere Textdatei zu schreiben. Die Quelldatei darf nur Buchstaben von A bis Z enthalten, Groß- und Kleinschreibung ist hierbei unwichtig.

Bereiten Sie eine Textdatei vor, in welcher sich lediglich aneinandergereihte Buchstaben befinden. Speichern Sie diese Datei ab (z.B. als Quelle.txt mit dem Inhalt „DiesIstEineQuelldatei“). Jetzt klicken Sie die Schaltfläche [Datei konvertieren] an. Es öffnet sich ein Öffnen-Dialog. Wählen Sie die zu codierende Datei aus, also beispielsweise die Datei „Quelle.txt“. Bestätigen Sie mit der Schaltfläche [Öffnen]. Es öffnet sich ein Speichern-Dialog. Jetzt vergeben Sie einen neuen Dateinamen, zum Beispiel „Ziel.txt“. Bestätigen Sie auch hier, aber diesmal mit Hilfe der Schaltfläche [Speichern]. Sie bekommen nebenstehende Erfolgsmeldung zu sehen, wenn alles geklappt hat. Um das Fenster mit der Erfolgsmeldung zu schließen, drücken Sie die Returntaste.

Es können auch hier verschiedene Fehlermeldungen auftreten. Beachten Sie hierzu bitte das gesonderte Kapitel „Fehlermeldungen“.

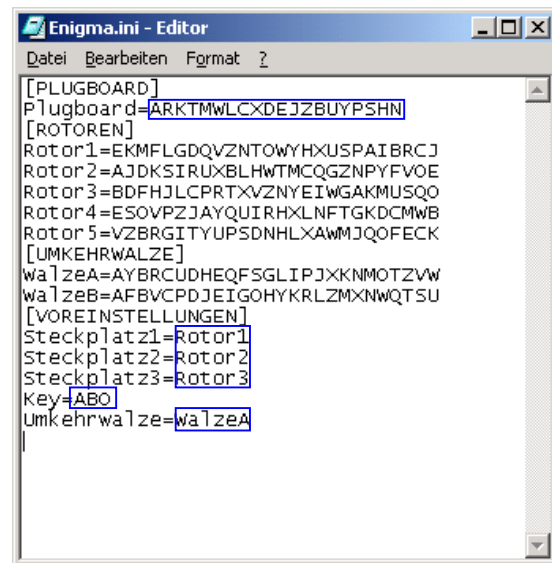


4.8 Voreinstellungen konfigurieren

Wenn Sie die Enigma starten, ist diese bereits mit Einstellungen versorgt. Sie haben die Möglichkeit diese Voreinstellungen zu verändern. Dazu befindet sich in dem Ordner der „Enigma.exe“ eine weitere Datei, sie trägt den Namen „Enigma.ini“.

Ini-Dateien können Sie mit einem Doppelklick ganz normal öffnen. Die „Enigma.ini“ ist in mehrere Abschnitte unterteilt, wie z.B. [ROTOREN]. Die Abschnitte bestehen wiederum aus Einträgen, wie z.B. „Rotor3=...“. Sie finden insgesamt eine textuelle Repräsentation der einzelnen Enigma-Komponenten wieder (vgl. Bild).

Die blau eingerahmten Bereiche der nebenstehenden Grafik bestimmen die Voreinstellungen der Enigma. Sie dürfen die rechte Seite eines Eintrages verändern (also nur nach einem Gleichheitszeichen!). Daraus folgt, dass Sie beispielsweise keine zusätzlichen Rotoren einfügen können.



Umkehrwalze:

Sie können bei der Umkehrwalze entweder „WalzeA“ oder „WalzeB“ eintragen.

Key:

Hinter dem Begriff „Key“ verbirgt sich die Anfangstellung der drei Rotoren. Sie können die Voreinstellung „ABO“ durch irgendeine dreibuchstäbige, großgeschriebene Buchstabenfolge ersetzen.

Rotoren:

Sie können die Rotoren in den Steckplätzen 1 bis 3 durch je eine der fünf Rotoren ersetzen. Sie dürfen allerdings keinen Rotor doppelt verwenden!

Plugboard:

Sie können das Plugboard komplett löschen oder es durch eine gerade Anzahl von maximal 20 unterschiedlichen, aber immer großgeschriebenen Buchstaben ersetzen. Vertauscht werden die Buchstaben immer paarweise. Beispiel: Plugboard=AKTE – Es werden A und K vertauscht und es werden T und E vertauscht. Alle anderen Buchstaben würden unverändert bleiben.

Achtung:

Es ist ebenfalls möglich die übrigen Einträge zu verändern. Allerdings wird dringend davon abgeraten, da eine korrekte Decodierung dann nicht mehr gewährleistet werden kann.

Ist die Enigma-Ini-Datei fehlerhaft oder nicht vorhanden, wird sie beim Starten des Programms neu angelegt.

5. Wiederanlaufbedingungen

Sollte Ihr Computer-System während der Arbeit mit der Enigma, beispielsweise durch einen Stromausfall, abstürzen, dann hat dies folgende Konsequenzen:

- Die aktuelle Chiffrierung ist verloren.
- Für die Sitzung vorgenommene, also individuelle Einstellungen sind verloren.
- In Dateien gesicherte Einstellungen, die Enigma-Datensätze (Dateiendung .dat), können neu eingelesen werden.
- Ist das System während einer Konvertierung abgestürzt, muss im Einzelfall geprüft werden, ob die Zieldatei vollständig gespeichert wurde.
- Die Ini-Datei (Enigma.ini) ist nicht beeinträchtigt worden.

6. Fehlermeldungen

Fehlermeldung	Fehlerursache	Behebungsmaßnahme
Es wurden die alten Einstellungen geladen, weil die Eingaben unvollständig sind. Bitte beachten Sie: (...)	Sie haben versucht unvollständige Einstellungen zu übernehmen. Häufige Fehlerursache: Die Anfangstellung der Rotoren wurde nicht mit Return bestätigt.	Lesen Sie nach, wie die Einstellungen korrekt vorgenommen werden und versuchen Sie es noch einmal.
Die Datei wird nicht gespeichert, weil die Eingaben unvollständig sind. Bitte beachten Sie: (...)	Sie haben versucht unvollständige Einstellungen in einer Datei zu sichern.	Vervollständigen Sie die Einstellungen und wiederholen Sie das Sichern der Einstellungen.
Speichern fehlgeschlagen.	Der Fehler ist außerhalb dieses Programms zu suchen.	Starten Sie Ihr System neu.
Öffnen fehlgeschlagen.	Der Fehler ist außerhalb dieses Programms zu suchen.	Starten Sie Ihr System neu.
Die Datei (...) ist fehlerhaft. Sie wird nicht geöffnet.	Entweder handelt es sich um eine Datei mit der Endung .dat, welche nicht zu dieser Enigma gehört; oder die Datei ist beschädigt worden.	Wenn es sich um einen beschädigten Enigma-Datensatz handelt, können Sie diesen nur noch mit gültigen Einstellungen überschreiben oder löschen.
Die Datei (...) existiert nicht. Sie wird nicht geöffnet.	Es wird versucht eine Datei zu öffnen, welche es aber nicht gibt.	Wiederholen Sie den Vorgang, aber geben Sie eine existierende Datei zum Öffnen an.
Die Datei (...) wird nicht geschrieben.	Sie haben einen Vorgang abgebrochen.	Bestätigen Sie die Meldung.
Die Datei (...) ist schreibgeschützt. Soll der Schreibschutz entfernt werden?	Sie versuchen eine schreibgeschützte Datei zu überschreiben.	Bestätigen Sie die Meldung mit „Ja“, wird der Schreibschutz entfernt und das Programm führt seine Aktion fort; anderenfalls bricht das Programm den Vorgang ab.
Soll die Datei: (...) überschrieben werden?	Sie versuchen eine Datei zu überschreiben, indem Sie beim Speichern eine existierende Datei ausgewählt haben, statt einen neuen Namen zu vergeben.	Bestätigen Sie die Meldung mit „Ja“, wird die Datei überschrieben, anderenfalls bricht das Programm den Vorgang ab.
Ihre ini-Datei ist Fehlerhaft. Sie wird neu angelegt und eingelesen.	Die Datei „Enigma.ini“ wurde mit ungültigen Werten verändert.	Bestätigen Sie die Meldung. Die falsche Ini-Datei wird durch eine funktionierende ersetzt.
Die Datei enthält ungültige Zeichen. Sie wird nicht konvertiert!	In der Textdatei befinden sich Sonderzeichen, Leerzeichen, Zahlen, oder Steuerzeichen.	Entfernen Sie alle ungültigen Zeichen aus der Textdatei. Speichern Sie die Datei ab und wiederholen Sie die Konvertierung.

II. Programmierhandbuch

1. Entwicklungskonfiguration

CPU	: PentiumIII mit 1,4 GHz
Arbeitsspeicher	: 1024 MB
Festplattenkapazität	: 80 GB, von Samsung
Monitor	: 17" - Monitor
Betriebssystem	: Windows XP, SP4
Entwicklungsumgebung	: Borland Delphi 7, Enterprise

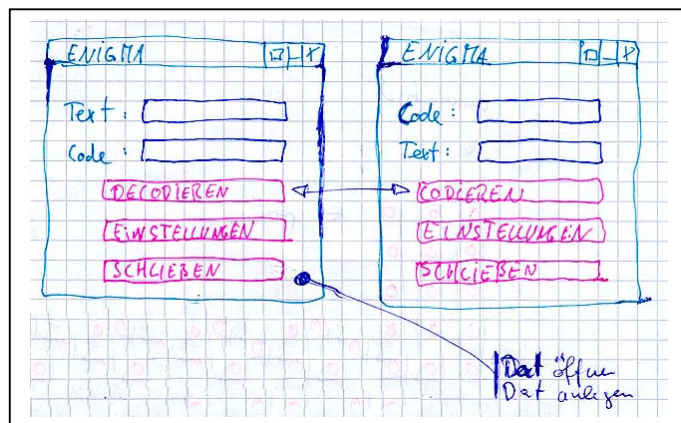
2. Problemanalyse und Realisation

2.1 Was soll implementiert werden? (Problemanalyse)

Um diese Frage zu beantworten wird das Interface entwickelt. Dazu wird die geforderte Funktionalität in grafische Elemente unterteilt und zwar mit der begleitenden Fragestellung: „Was soll der Benutzer machen können?“ Das Interface soll so entwickelt werden, dass falsche Eingaben, soweit möglich, vermieden werden.

2.1.1 Hauptformular

Nach Programmstart soll der Benutzer die Möglichkeit haben einen beliebigen Text einzugeben, um sofort das codierte Ergebnis zu gelangen (Editfelder). Will der Benutzer einen Text decodieren, dann muss er die Möglichkeit haben die Rotoren wieder auf Anfangsstellung zurückzudrehen [Decodieren / Codieren]. Darüber hinaus soll der Benutzer die Möglichkeit haben sämtliche Einstellungen verändern zu können. Dafür ist eine gesonderte Oberfläche vorgesehen [Einstellungen]. Die Aufgabenstellung verlangt weiterhin, dass der Benutzer eine Datei konvertiert in eine neue Datei schreiben können soll [Datei konvertieren]. Abschließend wird natürlich die Möglichkeit bereitgestellt das Programm ordnungsgemäß zu beenden [Schließen].

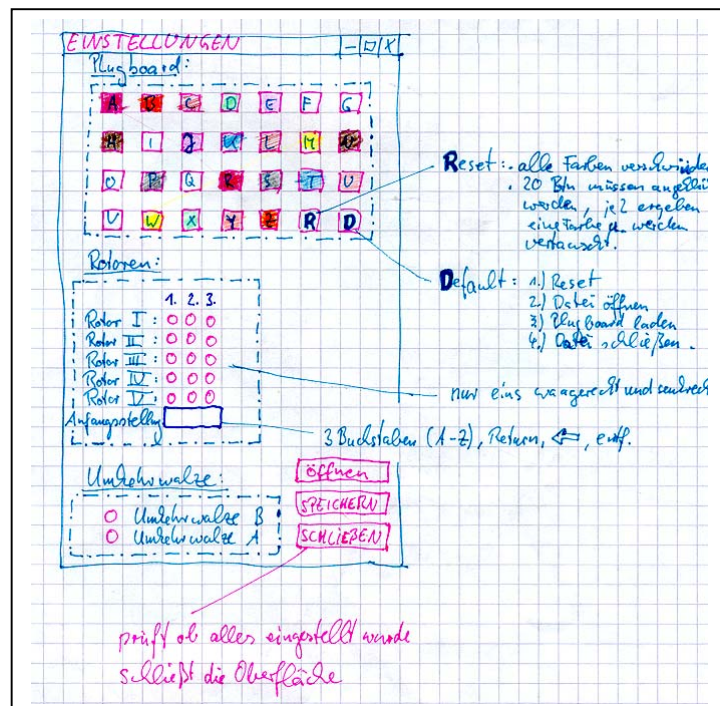


2.1.2 Formular Einstellungen

Die Programmoberfläche wird in vier logische Bereiche unterteilt:

1. Einstellen des Plugboards (PB):

Der Benutzer erhält die Möglichkeit Buchstaben miteinander zu vertauschen. Es sind verschiedene Realisierungen denkbar. Bei der hier skizzierten Version wird die Zusammengehörigkeit zweier Buchstaben visuell verdeutlicht: Gleiche Farben bedeuten gleiche Buchstaben. Will der Benutzer ein eigenes PB erstellen, so muss er auf



[Reset] drücken, um danach durch Anklicken von Buchstabenpaaren je eine Vertauschung vorzunehmen. Der Benutzer soll das PB auch auf Standardwerte zurücksetzen können [Default].

2. Wahl der Rotoren:

Aus 5 Rotoren sollen 3 Rotoren gewählt werden. Es muss dabei sichergestellt sein, dass drei Radio-Buttons angeklickt werden, aber in jeder Zeile, sowie in jeder Spalte höchstens ein Radio-Button ausgewählt wird. Dieses Problem wird gelöst, indem der Benutzer drei mal, aus erst 5, dann 4 und zuletzt aus 3 Radio-Buttons, wählen kann (vgl. Kap. I. 4.6.2 Rotoren ändern, Seite 7).

Die Anfangsstellung der drei Rotoren kann der Benutzer per Tastatur in ein Editfeld eingegeben. Das Editfeld lässt von vornherein nur gültige Eingaben zu.

3. Wahl der Umkehrwalze:

Für die Auswahl einer Umkehrwalze werden Radio-Buttons verwendet. Der Benutzer kann sich für die jeweils andere Umkehrwalze entscheiden.

4. Umgang mit den Einstellungen:

Dem Anwender soll weiterhin die Möglichkeit gegeben werden die neuen Einstellungen in einer Datei zu sichern und auch wieder von einer gesicherten Datei einzulesen [öffnen][speichern]. Außerdem müssen die neuen Einstellung auf Vollständigkeit geprüft werden (Korrekt sind die Einstellungen immer!) und dann auch wirksam werden. Dazu dient der Button [Schließen], welcher gleichzeitig das Formular Schließt.

2.2 Wie soll die Funktionalität hergestellt werden? (Realisierungsanalyse)

Um diese Frage zu beantworten werden für die verschiedenen Probleme je eine Funktionsbeschreibung angegeben und mögliche Ansätze zur Umsetzung beschrieben, gleichzeitig werden die damit verbundenen Datenstrukturen besprochen.

2.2.1 Datei Konvertieren

Funktionsbeschreibung :

Jeder Buchstabe einer ausgewählten Textdatei soll gelesen, anschließend codiert und schlussendlich in eine neue Textdatei geschrieben werden. Die Quelldatei darf nur aus gültigen Zeichen bestehen (A..Z, a..z). Vorhandene Dateien dürfen nicht beschädigt werden.

Ansätze zur Umsetzung :

Der Inhalt der Quelldatei kann in einen String geschrieben werden. Danach kann die Quelldatei geschlossen werden; sie wird nicht mehr benötigt. Der String könnte iterativ, also Buchstabenweise codiert werden und das Ergebnis kann Buchstabenweise in die Zieldatei geschrieben werden.

2.2.2 Plugboard

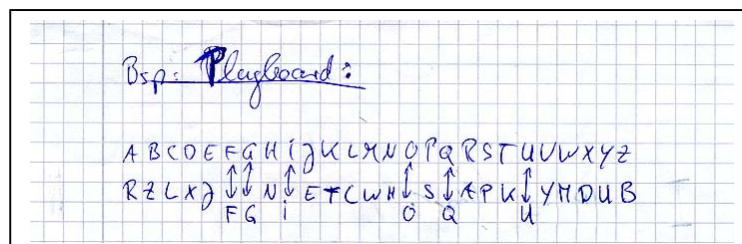
Funktionsbeschreibung :

Das Plugboard (PB) soll je zwei Buchstaben miteinander vertauschen. Es gibt also zu jedem Buchstaben einen Partner. Wird beispielsweise ein A mit einem V vertauscht, dann muss ebenfalls gelten, dass ein V ein A ergibt. Mindestens sechs Buchstaben bleiben dabei unvertauscht: Sie bilden sich auf sich selbst ab.

Ansätze zur Umsetzung :

Es kann sich bei dem Plugboard intern um eine art Liste handeln, die zwingend mit dem Datentyp char zu tun hat. Es wäre möglich eine lineare Liste zu programmieren, welche man beim Suchen eines Codes stets von vorne bis hinten durchlaufen muss. Bei dieser Überlegung wird es aber schwierig die Entsprechung $A=V$; $V=A$ zu kontrollieren.

Eine andere Möglichkeit ist es, den Datentyp Array zu verwenden. Statt Zahlen könnte man, einen Teilbereichstyp für die Indizes benutzen. Vielleicht ein Array mit den Indizes A bis Z und als Inhalt verwendet man ebenfalls Buchstaben von A bis Z. Bei der Entsprechung der Buchstaben muss nur darauf geachtet werden, dass bei einem A ein V eingetragen wird und bei einem V entsprechend das A. Das ließe sich mit zwei direkten Zugriffen sehr kontrollieren.



2.2.3 Steckplätze

Funktionsbeschreibung :

Es sollen drei Steckplätze mit je einem festen Rotor geben.

Ansätze zur Umsetzung :

Die naheliegendste Realisierung besteht darin drei feste Steckplätze zu implementieren. Steckplatz1 vom Typ TRotor, Steckplatz2 vom Typ TRotor und den dritten Steckplatz ebenso vom Typ TRotor. Bei dieser Implementierung gibt es allerdings einen großen Nachteil: Es muss ständig herausgefunden werden, welcher Rotor in welchem Steckplatz steckt, daraus folgt dann erst welcher Rotor überhaupt verwendet wird. Auch problematisch ist der umgekehrte Fragestellung: In welchem Steckplatz steckt eigentlich welcher Rotor? Mithilfe eines Case-Verteilers ist es zwar relativ leicht diese Abfragungen durchzuführen. Aber diese Informationen werden im Programmablauf sehr oft benötigt. Die permanente Prüfung ist schlichtweg zu Zeitaufwendig – und auch unhandlich.

Ein eleganterer Umsetzungsansatz wäre wohlmöglich ein Array mit drei Werten (repräsentieren Steckplätze) vorzusehen und dann sprichwörtlich in jeden Steckplatz einen Rotor „Reinzustecken“, also abzuspeichern. Sehr komfortabel kann mit dieser Version ermittelt werden, welche Rotoren verwendet werden. Außerdem kann das komplette Array übergeben werden, womit dann alle Rotoren bekannt wären. Auch bekannt ist die Zuordnung von Rotor zu Steckplatz und Steckplatz zu Rotor.

2.2.4 Rotoren

Funktionsbeschreibung :

Die Rotoren sind durch zyklische Ringe zu implementieren. Ein Rotor kann Buchstaben sowohl codieren, als auch decodieren. Nach jedem Chiffrierungsschritt muss der jeweilige Rotor „weitergedreht“ werden, damit bei mehrfacher, gleicher Eingabe unterschiedliche Ergebnisse erzielt werden.

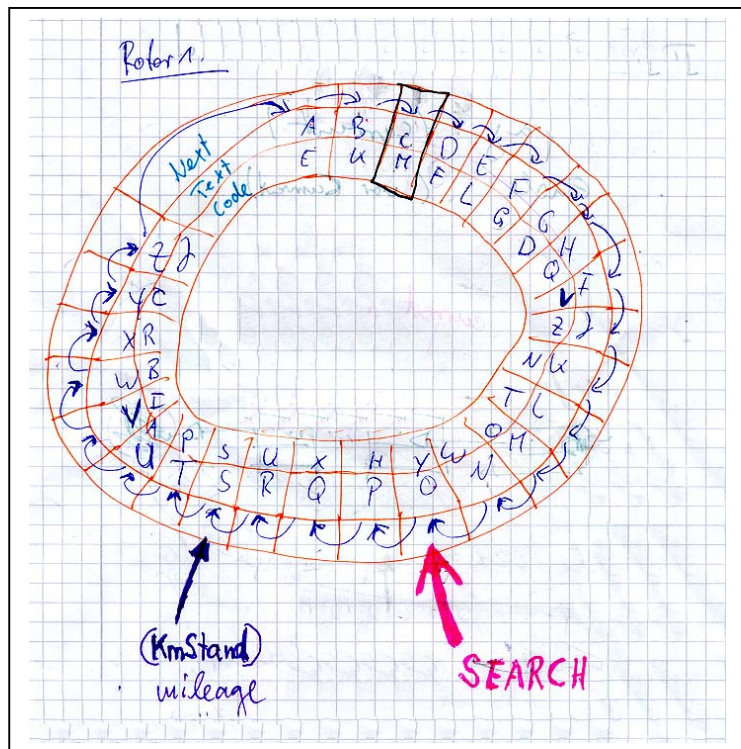
(Wie ein Rotor funktioniert lesen Sie im Benutzerhandbuch: Kap. I 4.1 Hintergrundinformationen)

Ansätze zur Umsetzung :

Die Infokomponente könnte aus zwei Elementen bestehen: Text und Code. Wird beispielsweise ein „C“ codiert (PB → Rotor → Umkehrwalze), dann wird der Einstiegspunkt „SEARCH“ solange weitergesetzt, bis im Feld „Text“ das gesuchte „C“ steht und das „M“ aus dem Feld „Code“ wird zurückgegeben.

Soll ein Buchstabe decodiert werden (Umkehrwalze → Rotor → PB), dann wird „SEARCH“ eben solange weitergesetzt, bis im Feld „Code“ der zu decodierende Buchstabe steht. Zurückgegeben wird in diesem Fall der Buchstabe im Feld „Text“.

Es ist grundsätzlich ein zweiter Einstiegspunkt in die Ringstruktur notwendig: „Mileage“ wird auf die gewünschte Anfangsstellung gesetzt und nach jedem Chiffrierungsvorgang (der beinhaltet codieren und decodieren) um ein Element weitergedreht. Zeigt „mileage“ wieder auf die Anfangsstellung, dann hat sich der Rotor einmal komplett gedreht. Der dahintersitzende Rotor kann sich um eine Stelle weiterdrehen („mileage“ von dem nächsten Rotor ein Element weitersetzen).



2.3 Wie wurde die Funktionalität hergestellt? (Realisierungsbeschreibung)

2.3.1 Grundsätzliches

Sämtliche Einstellungen sind immer gültig. Daraus folgt, dass die Enigma zu jedem Zeitpunkt codieren und decodieren kann.

Umkehrwalze:

Die Umkehrwalze ist ein spezielles Plugboard, bei dem sichergestellt sein muss, dass alle 26 Buchstaben immer vertauscht sind. Genau dann gibt es den Umkehreffekt.

Die Umkehrwalze wird also genauso behandelt, wie ein Plugboard. Allerdings gibt es eine zusätzliche Prüfung beim Initialisieren und beim Lesen des Eintrags aus der Ini-Datei.

FindComponent:

FindComponent ist eine Funktion von Delphi, genaueres kann also in der entsprechenden Hilfe nachgeschlagen werden. Hier lediglich ein Beispiel, um die Rotoren auf der Oberfläche entsprechend der internen Einstellungen zu setzen:

Deklaration:

Name : TRadioButton; (weitere...)

Quelltext:

```
for SP_Nr := 1 to 3 do                                // Steckplatz 1 bis 3 abarbeiten
begin
  Platz := IntToStr(SP_Nr);                            // Steckplatz als string
  Nr := IntToStr(Einst.Steckplatz[SP_Nr]);              // Welcher Rotor steckt im Steckplatz?
  Name := findComponent('rb_' + Platz + Nr) as TRadioButton; // Name d. Komponente herausfinden
  // Es resultiert der Name einer Komponente, z.B. „rb_24“ vom Komponententyp TRadioButton.
  // Rotor 4 ist in Steckplatz 2.
  Name.Checked := true; // entspricht der speziellen Anweisung : rb_24.Checked := true;
end;
```

2.3.2 Programmabläufe

Es werden die zum Verständnis notwendigen und damit grundlegenden und wichtigsten Abläufe in ihren Kernpunkten beschrieben.

Programm-Start (foMain):

- 1.) Aus der Ini-Datei werden alle Einstellungen gemäß den Einträgen in der Sektion „Voreinstellungen“ geladen. Diese Einstellungen werden auf Vollständigkeit und Richtigkeit geprüft. Sind Einstellungen fehlerhaft, dann wird die Ini-Datei neu angelegt.
- 2.) Alle Einstellungen werden gesetzt.
- 3.) Aus der Ini-Datei werden die nötigen Strings gelesen, um die Ringstruktur zu initialisieren.

Programm-Ende (foMain):

- 1.) Noch geöffnete Dateien werden -falls nötig- per Try-Finally – Block geschlossen.
- 2.) Die Ringstruktur wird in jedem Fall wieder freigegeben.

Datei Konvertieren (foMain):

- 1.) Öffnen Dialog (Pfad für Quelldatei)
- 2.) Schließen Dialog (Pfad für Zieldatei)
- 3.) Chiffrieren (in lgMain):
 - a. Rotoren auf Anfangsstellung setzen
 - b. Quelle einlesen (AnsiString: ~2³¹ Zeichen, bis 2 GB) (in unTypDatei)
 - i. öffnen,
 - ii. lesen,
 - iii. schließen
 - c. den String Buchstabenweise von vorne nach hinten auf gültige Zeichen prüfen und Codieren
 - d. Ergebnis in Zielstring schreiben
- 4.) wenn alles geklappt hat: Erfolgsmeldung ausgeben.

Formular „Einstellungen“ öffnen (in foMain):

- 1.) alte Einstellungen sichern
- 2.) Die Oberfläche mit den aktuellen Einstellungen vorbereiten (in foMain)
 - a. **show_Plugboard:**
Das PB-Array iterativ abarbeiten:
wenn A = A, Panel-Komponente p_A finden und neutrale Farbe setzen
wenn B = Z, Panel-Komponente p_B und p_Z finden und Farbe aus Farbarray setzen
„Z“ zum überspringen merken
Farbindex weitersetzen und...
mit C fortfahren...
 - b. **show_Rotoren:**
alle Rotoren sichtbar machen
iterativ die Steckplätze abarbeiten:
herausfinden, welcher Rotor in dem Steckplatz steckt
diesen Rotor auf der Oberfläche markieren (vgl. FindComponent, Kap. II.2.3.1)
 - c. **show_Umkehrwalze:**
prüfen welche Umkehrwalze aktuell verwendet wird und entsprechend auf der Oberfläche markieren.
- 3.) Oberfläche Einstellungen modal Anzeigen

Plugboard ändern (in foEinstellungen):

Alle PB Elemente (Panels) zeigen auf eine onClick-Procedur.

AnzClick = Die Anzahl der geklickten PB-Panels wird unit-global verwaltet.

merk1 = Der erste Buchstabe eines Buchstabenpaares wird unit-global verwaltet.

Farben = Farbwerte werden als Aufzählungstyp konstant und unit-global verwaltet

- 1.) AnzClick erhöhen
- 2.) Farbe bei jedem 2. Klick ändern (index entsprechend setzen)
- 3.) Sicherstellen, dass höchstens 20 Buchstaben vertauscht werden
- 4.) wenn AnzClick ungerade ist, handelt es sich um den ersten Klick
 - a. Komponente einfärben
 - b. Komponente unbenutzbar machen
 - c. Buchstabe merken
- 5.) wenn AnzClick gerade ist, handelt es sich um den zweiten Klick
 - a. Komponente einfärben
 - b. Komponente unbenutzbar machen
 - c. Beide Buchstaben im Array –intern– vertauschen

Einstellungen speichern (lgEinst):

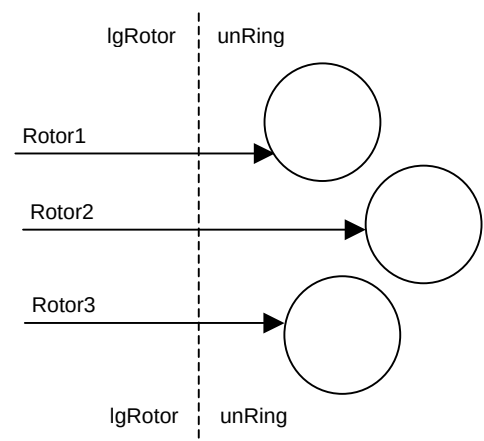
Die einzelnen Einstellungen sind in einem Record zusammengefasst. Der Record wird komplett in eine typerisierte Datei geschrieben (unTypDat). Der Dateityp ist TDatensatz, das ist eine Identität von dem Datentyp TEinstellungen (Identitäten werden in unTypen hergestellt).

Die Unit unDatTools stellt sämtliche Prüfroutinen zur Dateiverarbeitung bereit. (überschreiben verhindern, Schreibschutz erkennen und entfernen, Benutzer fragen, usw.)

Ringe initialisieren (in lgRotor):

Rotor1, Rotor2, Rotor3 = unit-globale Einstiegspunkte in die Ringstruktur

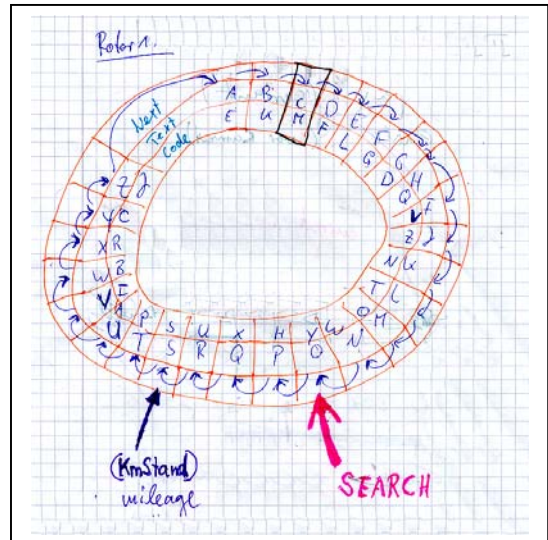
- 1.) Herausfinden, welcher Rotor steckt in welchem Steckplatz.
Folgende für jeden Steckplatz ausführen (3 mal):
- 2.) Den entsprechenden String aus der Ini-Datei lesen (unIniDatei)
- 3.) Den String Buchstabenweise in den entsprechenden Ring hinzufügen (in unRing):
 - a. Neues Element erstellen, **new()**
 - b. Buchstabe Hinzufügen
 - c. wenn Ring leer:
Element auf sich selbst zeigen lassen
 - d. wenn Ring Elemente hat:
Element einfügen
 - e. Zeiger auf neues Element an lgRotor zurückgeben



2.3.3 Chiffrierung

Problem: Weitersetzen:

Es hat sich gezeigt, dass die Vorüberlegungen zu den Rotoren, wie beschrieben (vgl. Kap: II. 2.2.4 Rotoren) nicht ganz richtig sind. Das zentrale Problem bei dem skizzierten Lösungsansatz ist die Tatsache, dass durch das Weitersetzen des Einstiegszeigers „mileage“, keine neuen Buchstabenpaare gebildet werden. Ein Buchstabe A, fünf mal hintereinander eingegeben, würde immer denselben chiffrierten Buchstaben ergeben. Konsequenterweise müsste man die Ringstruktur nach jeder Chiffrierung komplett umspeichern: Man müsste jeden Buchstaben aus dem inneren Ring (mit Code beschrieben) genau eine Position weiter abspeichern. Das wiederum ist schlichtweg Blödsinn und hat nichts mit einer guten Ringstruktur zu tun!



Lösung: Umstrukturierung:

Die Ringstruktur wurde deshalb um die Infokomponente „Text“, in der Zeichnung der mittlere Ring, verringert. Auch der Einstiegsplatz „mileage“ ist überflüssig.

Rotoren

Die Logik der Rotoren, das beinhaltet Codieren und Decodieren, initialisieren der Rotoren, auf Anfangstellung setzen etc. ist in der Unit IgRotoren realisiert worden. Die tatsächlichen Ringoperationen sind hingegen in der unRing realisiert worden.

Der Einstiegsplatz für einen Ring wird in IgRotor verwaltet.

Codierung eines E's (IgRotor):

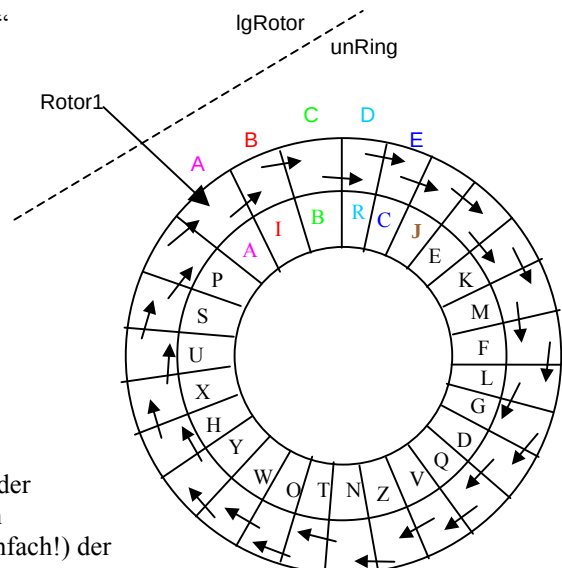
- 1.) Der globale Einstiegsplatz (Rotor1) wird lokal kopiert (run).
- 2.) Der lokale Zeiger wird nun E-mal weitergesetzt
- 3.) Rückgabewert ist der Buchstabe, auf den run aktuell zeigt. Es wurde der Buchstabe „C“ ausgezählt.

Decodierung eines C's (IgRotor):

- 1.) Der globale Einstiegsplatz (Rotor1) wird lokal kopiert (run).
- 2.) Der lokale Zeiger (run) wird nun solange weitergesetzt, bis dieser auf das gesuchte C zeigt.
- 3.) Die Schritte, bis zum C werden Alphabetisch mitgezählt.
- 4.) Der Rückgabewert ist der aktuelle Zählstand. Es sind genau E Schritte notwendig.

Rotor drehen (IgRotor):

- 1.) Es werden die aktuellen Einstiegsplätze mit der Anfangstellung (dreistelliger Key) verglichen
- 2.) Soll ein Rotor gedreht werden, dann wird (einfach!) der globale Einstiegsplatz auf die nächste Komponente gesetzt. Rotor1 zeigt nun nicht mehr auf A, sondern auf I. Bei der nächsten Codierung wird nun ab dem I gezählt. (Ein „E“ würde jetzt beispielsweise ein „J“ ergeben!)



2.4 Memory Check (Übersicht)

Veränderungen, sowie die Anzeige, beziehen sich auf den Rotor in diesem Steckplatz

Um einen Buchstaben manuell aus dem Ring zu entfernen bzw. wieder hinzuzufügen.

Alle Ringelemente werden gefüllt / freigegeben entspricht Programmstart und -ende.

Aktualisiert Plugboard, Umkehrwalze, Memofeld und die gelöschten Buchstaben werden aufgelistet.

Setzt die Rotoren auf Anfangstellung zurück.

Eingabefeld zum Codieren eines Buchstabens.

In Steckplatz 1 steckt zur Zeit der Rotor mit der Nr. 1. Die Einstiegsadresse in den jew. Ring ist angegeben. Sie ist stets die Erste „Self“ Adresse im Memofeld

Sollten zu Testzwecken einzelne Elemente aus einem Ring entfernt werden, dann werden Diese nach dem Aktualisieren hier angezeigt. Ring1 = Ring in Steckplatz 1

Aktuelles Plugboard und aktuelle Umkehrwalze.

Eine Codierung wird mit allen Zwischenschritten (untereinander) angezeigt.

1. Spalte:
Zur Orientierung wird das Alphabet in einer Schleife mit ausgegeben.
(Das nachzuvollziehen der Codierung / Decodierung der Enigma wird dadurch stark vereinfacht.)

2. Spalte (Info-Komponente):
Von oben nach unten gelesen ergeben sich die Buchstaben des Rotors.
In der **ersten Zeile** steht der Buchstabe, auf den gerade gezeigt wird. Beim Weiterdrehen des Rotors wird der Einstiegspunkt um eine Stelle weitergesetzt; das „I“ stünde dann an erster Stelle, alle Buchstaben rückten eine Zeile auf und das „A“ stünde ganz unten.

```
A -- A Self: $C32410 Next: $C3241C
B -- I Self: $C3241C Next: $C32314
C -- B Self: $C32314 Next: $C32428
D -- R Self: $C32428 Next: $C32434
E -- C Self: $C32434 Next: $C32440
F -- J Self: $C32440 Next: $C3232C
G -- K Self: $C32338 Next: $C32344
H -- M Self: $C32344 Next: $C32350
I -- F Self: $C32350 Next: $C3235C
J -- L Self: $C3235C Next: $C32368
K -- G Self: $C32368 Next: $C32374
L -- D Self: $C32374 Next: $C32380
...
Y -- S Self: $C323F8 Next: $C32404
Z -- P Self: $C32404 Next: $C32410
```

Self: \$123456
Das Ringelement mit der Infokomponente „A“ hat die Speicheradresse \$C32410_{HEX}.
Es handelt sich um die, mit einem new() – Aufruf allozierte, Adresse aus dem Heap-Speicher.
Vgl. II. 3.2 Rotoren / Ringe

Next: \$123456
In der Nextkomponente des Ringelementes A steht die Speicheradresse \$C3241C_{HEX}.
Weil das Element „I“ (Self) die Adresse \$C3241C hat, zeigt das Element „A“ auf das Element „I“.
Vgl. II. 3.2 Rotoren / Ringe

3. Datenstrukturen

3.1 Einstellungen

Der Datentyp TPlugboard besteht aus einem Array, in welchem einzelne Buchstaben abgespeichert werden. Das Array kann von „A“ bis „Z“ durchlaufen werden. Dieser Datentyp stellt sowohl das Plugboard, als auch die Umkehrwalze dar. Der Datentyp TSteckplatz ist ein Array und repräsentiert die drei Steckplätze der Enigma-Maschine. Zu jedem Steckplatz wird die Information festgehalten, welcher Rotor in welchem Steckplatz steckt. Mithilfe dieser Zuordnung ist es möglich festzustellen, welcher Rotor im Einzelfall angesprochen werden muss.

Die Anfgstellung der Rotoren wird in dem Datentyp Tkey festgehalten. Es handelt sich um einen dreistelligen String. Alle Teileinstellungen sind in dem Record TEinstellungen zusammengefasst.

```

Tabc = 'A'..'Z';
TPlugboard = array[Tabc] of char;

TSteckNr = 1..3;
TRotorNr = 1..5;
TSteckplatz = array[TSteckNr] of TRotorNr;

TKey = string[3];

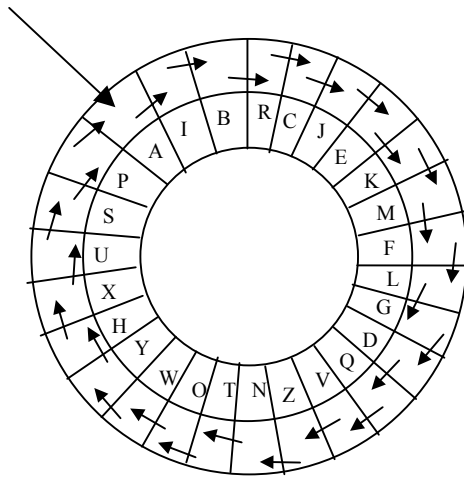
TEinstellungen = record
    Plugboard : TPlugboard;
    Steckplatz : TSteckplatz;
    Key : TKey;
    UmWalze : TPlugboard;
end;
```

Tplugboard:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
R	Z	L	X	J	F	G	N	I	E	T	C	W	H	O	S	Q	A	P	K	Y	V	M	D	U	B

3.2 Rotoren bzw. Ringe

Die Rotoren werden als Ringstruktur verwaltet. Ein Ring besteht aus Recordelementen, welcher wiederum aus einer Infokomponente (Info vom Typ TRotor) und einer Nextkomponente (Next vom Typ PRing) besteht. Die Infokomponente beinhaltet die Buchstaben der Rotoren. In der Nextkomponente steht die Speicheradresse des nächsten Ringabschnitts. Eine Grafik veranschaulicht den Aufbau eines Rings:

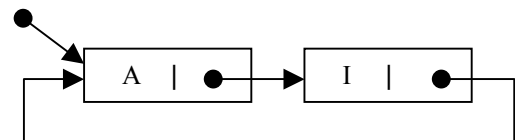


```
TRotor = 'A'..'Z';
```

```

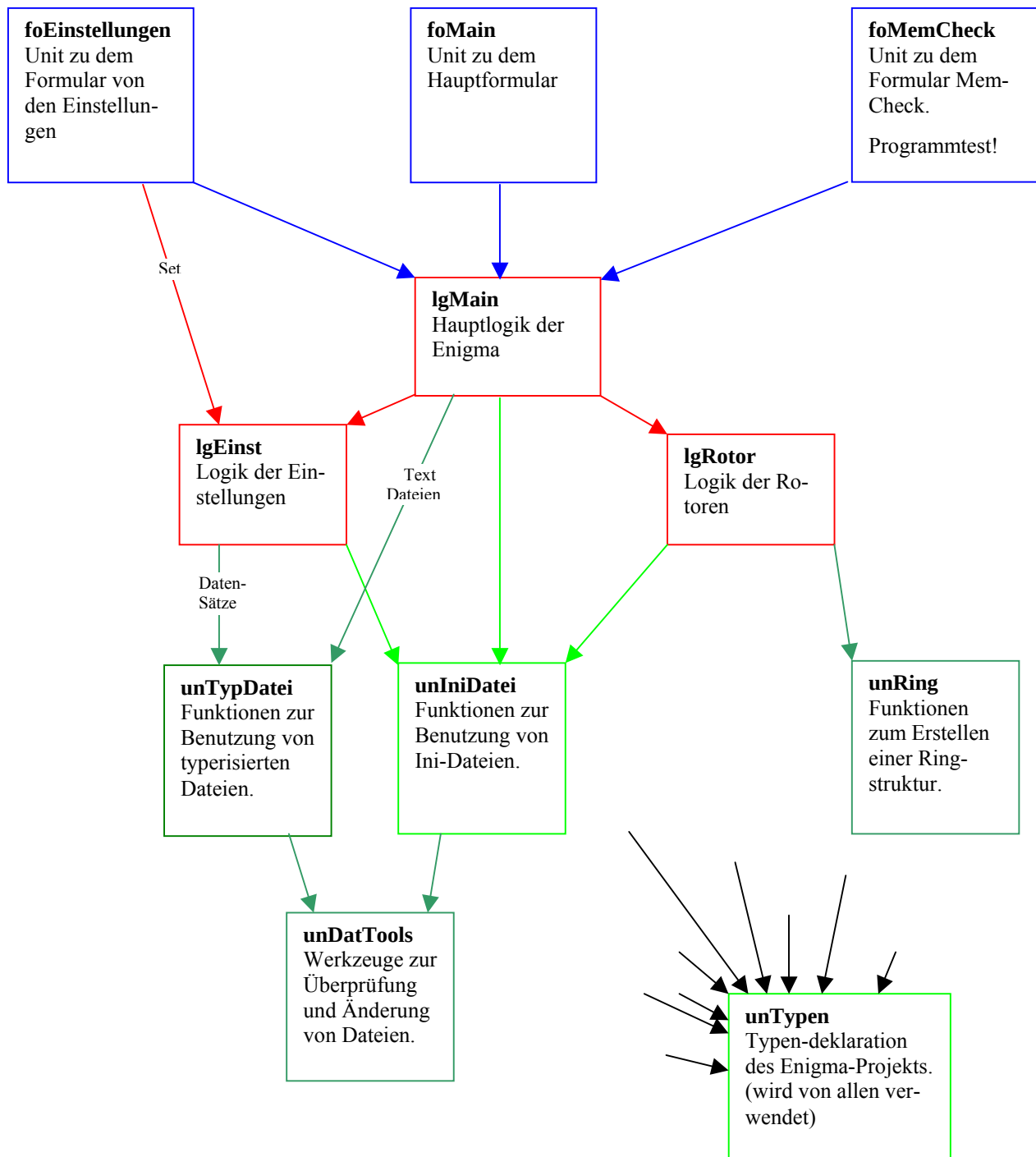
PRing = ^TRing;
TRing = record
    Next : PRing;
    Info : TRotor;
end;
```

Infokomponente	Nextkomponente
----------------	----------------



Das Element mit der Infokomponente „A“ zeigt auf das Element „I“.

4. Programmorganisationsplan



5. Programmtests

Datei Konvertieren		
Testfall	erwartetes Ergebnis	erzieltes Ergebnis
1. Text.txt in Code.txt konvertieren	Code.txt wird angelegt	entspricht erwartetem Ergebnis
2. Code.txt in reText konvertieren	reText.txt wird angelegt und besitzt denselben Inhalt, wie Text.txt	entspricht erwartetem Ergebnis
3. ReadOnly.txt in nicht_Überschrieben.txt konvertieren --> Nicht Überschreiben	Meldung: "Soll die Datei (...) überschrieben werden." Nein. Meldung: "Datei (...) wird nicht geschrieben!"	entspricht erwartetem Ergebnis
4. ReadOnly.txt in Überschrieben.txt konvertieren --> Überschreiben!	Meldung: "Soll die Datei (...) überschrieben werden." Ja. Datei wird mit neuem Inhalt überschrieben.	entspricht erwartetem Ergebnis
5. ReadOnly.txt in RO_Leer.txt konvertieren --> Schreibschutz nicht entfernen.	Meldung: "Soll die Datei (...) überschrieben werden." Ja. Meldung: "Datei (...) ist Schreibgeschützt, soll der Schreibschutz entfernt werden?" Nein. Datei wurde nicht geschrieben.	entspricht erwartetem Ergebnis
6. ReadOnly.txt in RO_Leer.txt konvertieren --> Schreibschutz entfernen.	Meldung: "Soll die Datei (...) überschrieben werden." Ja. Meldung: "Datei (...) ist Schreibgeschützt, soll der Schreibschutz entfernt werden?" Ja. Datei wurde mit neuem Inhalt überschrieben. Der Schreibschutz ist entfernt worden.	entspricht erwartetem Ergebnis
7. myself.txt in myself.txt kodieren.	Meldung: "Soll die Datei (...) überschrieben werden." Ja. Datei wird mit neuem Inhalt überschrieben.	entspricht erwartetem Ergebnis
8. KM-Zähler Test: 26x26x26 = 17576 Buchstaben. Die Datei AAA20k.txt besteht aus 20.000 A's, sie wird in die Datei AAA20k_Code.txt kodiert.	Nach genau 17576 Buchstaben wiederholt sich die kodierte Buchstabenfolge der AAA20k_Code -- Datei.	entspricht erwartetem Ergebnis
Einstellungen Sichern /Laden		
Testfall	erwartetes Ergebnis	erzieltes Ergebnis
1. anlegen von einst.dat	einst.dat wird erstellt	entspricht erwartetem Ergebnis
2. überschreiben von überschreib_mich.dat	Meldung: "Datei (..) existiert bereits, soll Sie überschrieben werden?" Ja. Datei wurde überschrieben.	entspricht erwartetem Ergebnis
3. überschreiben von Read_Only.dat	Meldung: "Datei (..) existiert bereits, soll sie überschrieben werden?" Ja. "Datei (..) ist Schreibgeschützt, soll SS entfernt werden?" Ja. SS ist entfernt worden, die Datei wurde überschrieben.	entspricht erwartetem Ergebnis

4. Laden von überschreib_mich.dat	Die Einstellungen werden geladen und im Fenster "Einstellungen" korrekt angezeigt.	entspricht erwartetem Ergebnis
5. Fehlerhafte Einstellungen sichern. (Bsp: Anfangstellung nicht mit Return bestätigt.)	Meldung: "Datei wird nicht gespeichert, weil die Eingaben unvollständig sind. Bitte beachten Sie (..)" Die Einstellungen können korrigiert werden.	entspricht erwartetem Ergebnis

Einstellungen vornehmen

Testfall	erwartetes Ergebnis	erzieltes Ergebnis
1. Fehlerhafte Einstellungen übernehmen (Bsp: Anfangstellung nicht mit Return bestätigt.)	Meldung: "Es wurden die alten Einstellungen geladen, weil die Eingaben unvollständig sind. Bitte beachten Sie (..)" Die Einstellungen können nicht mehr korrigiert werden. Das Fenster "Einstellungen" ist geschlossen.	entspricht erwartetem Ergebnis
2. Fehlerhafte Einstellungen abbrechen.	Das Fenster wird geschlossen, die alten Einstellungen werden verwendet.	entspricht erwartetem Ergebnis
3. Drei (ungerade Anzahl) Stecker im Plugboard getauscht. --> Übernehmen	Keine Meldung! Die korrekt vertauschten Buchstaben wurden vertauscht. Der "Partnerlose" wurde ignoriert und wird nicht vertauscht.	entspricht erwartetem Ergebnis
4. veränderte (aber vollständige) Einstellungen Übernehmen.	Das Fenster wird geschlossen, die neuen Einstellungen werden verwendet.	entspricht erwartetem Ergebnis
5. veränderte (aber vollständige) Einstellungen Abbrechen.	Das Fenster wird geschlossen, die alten Einstellungen werden verwendet.	entspricht erwartetem Ergebnis

Ini-Datei

Testfall	erwartetes Ergebnis	erzieltes Ergebnis
1. Es existiert keine "Enigma.ini" in dem selben Verzeichnis der Enigma.exe.	Eine korrekte Enigma.ini wird stillschweigend angelegt.	entspricht erwartetem Ergebnis
2. Plugboard hat mehr als 20 Buchstaben	Meldung: "Ihre Ini-Datei ist fehlerhaft. Sie wird neu angelegt und eingelesen." Programm startet mit den Einstellungen aus der überschriebenen Ini-Datei.	entspricht erwartetem Ergebnis
3. Plugboard hat keine gerade Anzahl von Buchstaben	Meldung: "Ihre Ini-Datei ist fehlerhaft. Sie wird neu angelegt und eingelesen." Programm startet mit den Einstellungen aus der überschriebenen Ini-Datei.	entspricht erwartetem Ergebnis
5. Die Rotoren bestehen nicht aus 26 unterschiedlichen Buchstaben.	Meldung: "Ihre Ini-Datei ist fehlerhaft. Sie wird neu angelegt und eingelesen." Programm startet mit den Einstellungen aus der überschriebenen Ini-Datei.	entspricht erwartetem Ergebnis
6. Doppelte Rotoren oder falsche Rotoren in den Voreinstellungen (z.B. "Ro4" oder "Rotor6")	Meldung: "Ihre Ini-Datei ist fehlerhaft. Sie wird neu angelegt und eingelesen." Programm startet mit den Einstellungen aus der überschriebenen Ini-Datei.	entspricht erwartetem Ergebnis

Enigma

7. Falscher Key in den Voreinstellungen (er ist nicht dreistellig oder besteht nicht aus Großbuchstaben)	Meldung: "Ihre Ini-Datei ist fehlerhaft. Sie wird neu angelegt und eingelesen." Programm startet mit den Einstellungen aus der überschriebenen Ini-Datei.	entspricht erwartetem Ergebnis
8. Die Umkehrwalze ist nicht "WalzeA" oder "WalzeB"	Meldung: "Ihre Ini-Datei ist fehlerhaft. Sie wird neu angelegt und eingelesen." Programm startet mit den Einstellungen aus der überschriebenen Ini-Datei.	entspricht erwartetem Ergebnis

Speicherverwaltung

Die folgenden Testfälle beziehen sich auf das Formular „Memory Check“ (siehe Kap. XXX).
(im Hauptformular Btn_MemoryCheck auf visible = true setzen.)

Testfall	erwartetes Ergebnis	erzieltes Ergebnis
1. Ist der Ring richtig verkettet?	Für jeden Eintrag i muss gelten: $Next_{(i)} = Self_{(i+1)}$. Die letzte Adresse ($Next_{(last)}$) muss dabei <i>immer (!)</i> auf die Adresse des ersten Ringelements ($Self_{(first)}$) zeigen.	entspricht erwartetem Ergebnis
2. Wird der Speicher wieder freigegeben? Einen Eintrag löschen, dazu in dem Editfeld „Löschen“ z.B. ein „C“ eingeben.	Meldung: „C gelöscht.“ Das „C“ wird aus dem Ring (=der zweiten Spalte) gelöscht. Beim Anzeigen der Ringelemente wird das C ausgelassen (erste Spalte). Die Next-Adresse von „B“ (Vorgänger) ist nun identisch mit der Self-Adresse von „D“ (Nachfolger). Zähler verringert sich um 1 (dispose).	entspricht erwartetem Ergebnis
3. Wird der Speicher für die Ringelemente richtig alloziiert? Einen Eintrag hinzufügen, dazu in dem Editfeld „Hinzufügen“ z.B. ein „C“ eingeben. Es können nur die Buchstaben hinzugefügt werden, die fehlen!	Meldung: „C wieder hinzugefügt.“ Das „C“ wurde an der nächsten Position von dem aktuellen Einstiegspunkt in den Ring (zweite Spalte) eingefügt. In der ersten Spalte wird das „C“ wieder angezeigt. $Next_{(i)} = Self_{(i+1)}$ gilt. Zähler erhöht sich um 1 (new).	entspricht erwartetem Ergebnis
4. Schrittweise entfernen eines ganzen Rings. Im Editfeld „Löschen“ das gesamte Alphabet eingeben (jeden Buchstaben mit Return bestätigen).	Meldung für jeden Buchstaben. Die Liste wird immer kürzer, die Ringstruktur bleibt dabei stets geschlossen. Ist nur noch ein Buchstabe vorhanden, zeigt dieser auf sich selbst. Ist kein Buchstabe mehr vorhanden, dann ist die Einstiegsadresse #0 (NIL).	entspricht erwartetem Ergebnis
5. Programm-Start simulieren: „Alle Füllen“	Alle drei Ringe werden gelöscht und komplett neu angelegt. Der Zähler steht bei $26 \times 3 = 48$.	entspricht erwartetem Ergebnis
6. Programm-Ende simulieren: „Alle Freigeben“	Alle drei Ringe werden gelöscht. Der Zähler steht auf 0. Die Einstiegspunkte sind alle drei #0 (NIL).	entspricht erwartetem Ergebnis

6. Änderungen...

6.1 ...am Quellcode

Datum	Name	Unit	Procedur / Funktion	Änderung
13.05.05	R. Räthel	-- Compilereinstellungen --		Optimierung = Aus Erweiterte Syntax = Aus Bereichsprüfung = Ein I/O-Prüfung = Ein Überlaufprüfung = Ein.
13.05.05	R. Räthel	foEinstellungen	saveClick (P)	showmessage statt messageDlg verwendet.
13.05.05	R. Räthel	foMemCheck	ed_Pointer2KeyPress (P)	vor ins_Ringelemente Variable und Zuweisungsoperator eingefügt.
13.05.05	R. Räthel	foEinstellungen	ed_KeyKeyPress (P)	untere Grenze der lokalen Variable „l“ (länge) auf 0 gesetzt, statt auf 1.
13.05.05	R. Räthel	unRing	unit-global	globale Variable "Current" entfernt.

6.2 ...an der Dokumentation

Datum	Name	Kapitel	Änderung
14.05.05	R. Räthel	5. Programmtests	Test-Tabelle „Speicherverwaltung“ hinzugefügt
14.05.05	R. Räthel	2.4 Memory Check (Übersicht)	Neues Kapitel eingeführt
14.05.05	R. Räthel	3.2 Rotoren bzw. Ringe	Grafik angepasst und hinzugefügt.
22.05.05	R. Räthel	2. Problemanalyse	komplett überarbeitet, diverse Kapitel eingefügt, Skizzen eingefügt
15.05.05	R. Räthel	6. Änderungen...	Neues Kapitel eingeführt
22.05.05	R. Räthel	Inhaltsverzeichnis	Inhaltsverzeichnis angepasst